Docket No.: END920010073US1

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

| | | |
|---|---|---|
| In Re Patent Application of: James R. Doran et al | | |
| Group Art Unit: 2194 | : | IBM Corporation |
| Examiner: Lechi Truong | : | Intellectual Property Law |
| Serial No.: 10/037,175 | : | Department SHCB/040-3 |
| Filed: 11/09/2001 | : | 1701 North Street |
| Title: ENTERPRISE DIRECTORY SERVICE | : | Endicott, New York 13760 |

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**APPEAL BRIEF**

Dear Sir:

Appellants hereby submit a revised Appeal Brief in response to a Notification of Non-Compliant Appeal Brief dated 01/17/2007. The Evidence appendix and Related Proceedings

appendix sections are added with an indication of "None".

**(i) REAL PARTY IN INTEREST**

The real party in interest is International Business Machines Corporation, a corporation of New York, with a place of business at Armonk, NY 10504.

**(ii) RELATED APPEALS AND INTERFERENCES**

There are no related appeals or interferences with which the undersigned is aware.

**(iii) STATUS OF CLAIMS**

Claims 1 - 22 are pending in the present application. Claims 1 - 22 have all been finally rejected and are the subject matter of this appeal.

**(iv) STATUS OF AMENDMENTS**

An amendment after Final under 37 C.F.R. 1.116 was filed 11/15/2005. In the Advisory Action dated 12/29/2005, the Examiner indicates that for purposes of appeal, this amendment will be entered.

**(v) SUMMARY OF CLAIMED SUBJECT MATTER**

Appellants' invention relates to a unique apparatus, method, system, computer program product, and deploying method

for providing directory service to a user application. Although directory service capability for individual users to, for example, look up names in a business directory are well known, there is a need within a business enterprise for various software applications to access the same directory type of information. The present invention directory service on a server computer is able to accept queries from user applications in a plurality of programming languages.

According to Appellants' independent claim 1, an enterprise directory service apparatus has a data store having a plurality of directory entries (specification page 8, lines 8 - 21). A web server has an API (application programming interface) coupled to the data store (page 9, lines 1 - 14). The API sends a query to the data store and receives a directory entry (page 9, lines 16 - 18 and page 11, lines 20 - 21). The enterprise directory apparatus also has a wrapper coupled to the API (page 9, lines 19 - 25). The wrapper is adapted for accepting the query from a user application in a plurality of programming languages (page 10, lines 1 - 22, page 11, lines 12 - 15, and page 13, lines 11 - 14).

Appellants' other independent claims 10, 18, 20, and 22, further require that the received directory entry be sent to the user application (page 11, lines 21 - 23).

**(vi) GROUNDS OF REJECTION**

There are three grounds of rejection. Clams 1, 2, 4, 5, 7

- 11, 13, 15, 16 - 18, and 20 - 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kirkwood (U.S. Patent 6,665,662) in view of Lippert (U.S. Patent 6,356,906).

Claims 3 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kirkwood in view of Lippert and further in view of Kumar (U.S. Patent 6,343,287).

Claims 6, 14, and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kirkwood in view of Lippert and further in view of Coden (U.S. Patent 5,873,080).

Appellants argue below that all of the independent claims 1, 10, 18, 20, and 22 are allowable over the first ground of rejection. All of the dependent claims are therefore also allowable and no argument is needed under the second and third grounds of rejection.

**(vii) ARGUMENT**

Independent claims 1, 10, 18, 20, and 22 are patentable under 35 U.S.C. 103(a) over the prior art and particularly U.S. Patents 6,665,662 (Kirkwood) in combination with U.S. Patent 6,356,906 (Lippert).

The combination of Lippert with Kirkwood does not describe all of the required elements or steps of Appellants' claims 1, 10, 18, 20, and 22. Appellants therefore respectfully disagree with this ground of rejection and offer the following arguments in support thereof.

Appellants' independent claims require a web server having an API (application programming interface) coupled to a data store for sending a query to the data store and receiving a directory entry. Appellants' independent claims also require a wrapper coupled to the API adapted for accepting the query from a user application in a plurality of programming languages. Examples of such programming languages are given in Appellants' specification page 11 as Java, LOTUS SCRIPT, REXX, and "C".

It is well known that programming languages are different from and distinguishable over query languages, such as SQL or markup languages such as HTML or XML. For example, programming languages are used by programmers to create executable applications. Query languages and markup languages are not used this way. One of ordinary skill would not be confused and could easily determine whether a language is a programming language, a query language, or a markup language.

As noted by the examiner in the Final Office Action dated 09/28/2005, page 2, last paragraph, Kirkwood does not describe a wrapper adapted for accepting a query in a plurality of programming languages. In fact, Kirkwood's server side adapter accepts and passes only XML documents. See Kirkwood, column 22, lines 5 - 8, and 21 - 27. XML is a markup language, for example, see Lippert, column 1, lines 40 - 44. XML is not a programming language.

Lippert teaches that his query in the SQL query language is wrapped in a markup language, such as XML, see column 2, lines

25 - 28, and column 25, lines 63 - 64. Lippert also performs this wrap step at the client side, not at the server. See his FIG. 2(a), FIG. 3, and column 6, lines 26 - 48.

Lippert then further wraps at the client side his XML wrapped query with a transport protocol, such as HTTP (Hyper Text Transport Protocol), in column 6, line 49 - column 7, line 25. This twice wrapped query is then sent from his client computer to his server computer. Lippert merely states in column 7, lines 29 - 30, that his "server processes the query and returns a suitable response", without any further details.

Lippert then describes wrapping this response with a markup language, such as XML, and then further wrapping with a transport protocol, such as HTTP (column 7, line 56 - column 8, line 67). It is then sent from his server computer to his client computer.

Appellants' independent claims require a web server having an API coupled to a data store and a wrapper coupled to the API adapted for accepting a query from a user application in a plurality of programming languages. Although Lippert, as noted above, describes wrapping in a markup language and wrapping in a transport protocol, Lippert does not describe how his server accepts and processes a query. He does state in column 10, lines 23 - 35, that his invention can be used with any database query language and any markup language, he does not describe, suggest, or imply that a query in a plurality of programming languages can be accepted. In fact, in his statement of advantages of his invention in column 10, line 61 - column 11,

line 4, Lippert notes that various clients and servers having inherent database capability such as SQL (a query language but not a programming language) can interact because his queries and responses are wrapped with transport protocols and markup languages understood by the other clients and other servers. However, he does not describe or suggest anything about accepting a query in a plurality of programming languages as required by Appellants' independent claims.

Neither Kirkwood, nor Lippert, nor any combination thereof, describe this required element of Appellants' invention. Appellants' independent claims are therefore allowable over Kirkwood in view of Lippert. Rejection of these claims and the claims dependent therefrom under 35 U.S.C. 103(a) is in error.

In view of the above, Appellants respectfully request that the Board reverse the Examiner's final rejection of all of the claims on appeal, and allow these claims.

Respectfully submitted,

Dated: 01/24/07

By: /John Pivnichny/
John R. Pivnichny
Reg. No. 43,001

Telephone: (607)429-4358
Fax: (607)429-4119

**(viii) CLAIMS APPENDIX**

1. An enterprise directory service apparatus, comprising:

a data store having a plurality of directory entries;

a web server having an API coupled to said data store, for sending a query to said data store and receiving a directory entry; and

a wrapper coupled to said API adapted for accepting said query from a user application in a plurality of programming languages.

2. The apparatus of claim 1, wherein said data store is a relational database.

3. The apparatus of claim 1, wherein said data store is an LDAP data store.

4. The apparatus of claim 1, wherein said web server has a plurality of API coupled to said data store, each API adapted to send said query to said data store and receive one of said plurality of directory entries.

5. The apparatus of claim 4, further comprising a plurality of wrappers each said wrapper coupled to one or more of said

plurality of API, and each said wrapper adapted to accept said query from one of a plurality of user applications.

6. The apparatus of claim 5, further comprising an API locator on said web server for selecting one of said plurality of API in response to said query from said one of said plurality of said user applications.

7. The apparatus of claim 1, wherein said API is adapted to receive one of said plurality of directory entries from said data store and send said one of said directory entries to said user application.

8. The apparatus of claim 7, wherein said API is adapted to send said one of said directory entries to said user application through said wrapper.

9. The apparatus of claim 7, wherein said API is adapted to receive said one of said plurality of directory entries in response to said query.

10. A method of providing directory service to a user application, said method comprising the steps of:

providing a data store having a plurality of directory entries;

providing a web server having an API coupled to said data store and a wrapper adapted to accept queries in a plurality of programming languages, coupled to said API;

receiving at said wrapper a query from a user application, and in response thereto sending said query from said wrapper to said API and thereafter to said data store; and

receiving at said API a directory entry from said data store in response to said query, and sending said directory entry to said user application.

11. The method of claim 10, wherein said data store is provided as a relational database.

12. The method of claim 10, wherein said data store is provided as a LDAP data store.

13. The method of claim 10, wherein said web server is provided having a plurality of API coupled to said data store, each API adapted to send said query to said data store and receive one of said plurality of directory entries.

14. The method of claim 13, further comprising the step of providing an API locator coupled to said wrapper and said

plurality of API for determining to which one of said plurality of API said wrapper should send said query.

15. The method of claim 13, further comprising the step of providing a plurality of wrappers, each said wrapper coupled to one or more of said plurality of API, and each said wrapper adapted to accept said query from one of a plurality of user applications.

16. The method of claim 10, further comprising the step of receiving one of said plurality of directory entries from said data store and sending said one of said directory entries to said user application.

17. The method of claim 16, further comprising sending said one of said directory entries to said user application through said wrapper.

18. A computer system for providing enterprise directory service, said system comprising:

means for providing a data store having a plurality of directory entries;

means for providing a web server having an API coupled to said data store and a wrapper adapted to receive queries in a plurality of programming languages, coupled to said API;

means for receiving at said wrapper a query from a user application, and in response thereto sending said query from said wrapper to said API and thereafter to said data store; and

means for receiving at said API a directory entry from said data store in response to said query, and sending said directory entry to said user application.

19. The system of claim 18, further comprising an API locator on said web server for selecting said API in response to said query from said user application.

20. A computer program product for instructing a processor to provide enterprise directory service, said computer program product comprising:

a computer recordable medium:

first program instruction means for providing a data store having a plurality of directory entries;

second program instruction means for providing a web server having an API coupled to said data store and a wrapper adapted to receive queries in a plurality of programming languages, coupled to said API;

third program instruction means for receiving at said wrapper a query from a user application, and in response thereto sending said query from said wrapper to said API and thereafter to said data store; and

fourth program instruction means for receiving at said API a directory entry from said data store in response to said query, and sending said directory entry to said user application; and wherein

all said program instruction means are recorded on said medium.

21. The computer program product of claim 19, further comprising fifth program instruction means for providing a wrapper coupled to said API for receiving said query from said user.

22. A method of deploying a directory service to a client, comprising the steps of:

providing data storage service including a data store having a plurality of directory entries;

providing a web service, said service capable of serving up web pages and having an API coupled to said data store and a wrapper adapted to receive queries in a plurality of programming languages, coupled to said API;

receiving at said wrapper a query from a client application, and in response thereto sending said query from said wrapper to said API and thereafter to said data store; and

receiving at said API a directory entry from said data store in response to said query, and sending said directory entry to said client application.

**(ix) EVIDENCE APPENDIX**

None.

**(x) RELATED PROCEEDINGS APPENDIX**

None.